**--1. Create a trigger on AmbulatoryVisits to raise exception on inserting new records with Provider\_Id and VisitDepartmentID which doesn't exist in the hospital.**

create or replace function av\_trig1()

returns trigger

Language PLPGSQL

as $$

begin

if (new."Provider\_ID">40 or new."VisitDepartmentID">12) then

raise exception

'ERROR: Provider\_ID or Visit department does not exist!';

else return new;

end if;

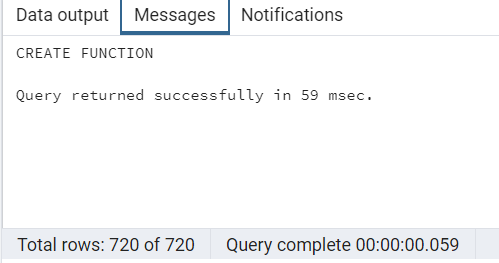
end $$;

create or replace trigger av\_trig1

before insert on "AmbulatoryVisits"

for each row

execute procedure av\_trig1();



A screenshot of a computer

Description automatically generated with medium confidence

**--Here, new record is entered with wrong ProviderID which shows the error and doesn't allow to**

**--enter that record.**

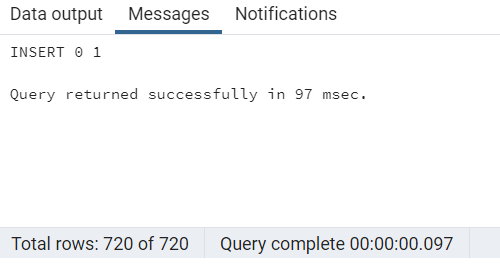
insert into "AmbulatoryVisits" ("Visit\_ID", "Provider\_ID","VisitDepartmentID") values(12345,42,12)

Graphical user interface, text, application, email

Description automatically generated

**--Here, new record is entered with correct ProviderID and VisitDepartmentID,and it is entered sucessfully.**

insert into "AmbulatoryVisits" ("Visit\_ID", "Provider\_ID","VisitDepartmentID") values(12346,35,12)



**-Both the records must be deleted for not to affect other outputs.**

delete from "AmbulatoryVisits" where "Visit\_ID" between 12345 and 12346

Graphical user interface, text, application, email

Description automatically generated

**--2. Create a trigger on the AmbulatoryVisit for displaying message while inserting new record**

**--for any abnormal blood pressure record.**

create or replace function av\_trig2()

returns trigger

Language PLPGSQL

as $$

begin

if new."BloodPressureSystolic"<90 then

perform 'Alarming: Low Systolic Blood Pressure';

END IF;

if new."BloodPressureSystolic">120 then

raise notice 'Alarming: High Systolic Blood Pressure';

END IF;

if new."BloodPressureDiastolic"<60 then

raise notice 'Alarming: Low Diastolic Blood Pressure';

END IF;

if new."BloodPressureDiastolic">80 then

raise notice 'Alarming: High Diastolic Blood Pressure';

END IF;

return new;

end

$$;

create or replace trigger av\_trig2

after insert on "AmbulatoryVisits"

for each row

execute procedure av\_trig2();

--Here, new record is entered with abnormal blood pressure.

Graphical user interface, text, application, email

Description automatically generated

insert into "AmbulatoryVisits"("Visit\_ID","BloodPressureSystolic","BloodPressureDiastolic")

values (12341,89,85)

A screenshot of a computer

Description automatically generated with medium confidence

**--Both the records must be deleted for not to affect other outputs.**

delete from "AmbulatoryVisits" where "Visit\_ID"=12341

Graphical user interface, text, application, email

Description automatically generated

**--3. Create a trigger on ReadmissionFlag when a patient gets readmitted.**

create or replace function rr\_trig3()

returns trigger

Language PLPGSQL

as $$

begin

if (new."ReadmissionFlag"=1) then

raise exception

'Message: One Patient got readmitted';

end if;

return new;

end $$;

create or replace trigger rr\_trig3

before update on "ReAdmissionRegistry"

for each row

execute procedure rr\_trig3();

A screenshot of a computer

Description automatically generated with medium confidence

update "ReAdmissionRegistry" set "ReadmissionFlag" =1 where "Patient\_ID"=10

Graphical user interface, text, application

Description automatically generated

**--4. Count the number of expired patients due to each illness type.**

select c."PrimaryDiagnosis" , count (d."Patient\_ID") as "Expired\_Patient"

from "Discharges" as d

Join "PrimaryDiagnosis" as c on c."Diagnosis\_ID"= d."Diagnosis\_ID"

Join "DischargeDisposition" as dd on d."Discharge\_ID" = dd."Discharge\_ID"

where dd."DischargeDisposition" = 'Expired'

group by c."PrimaryDiagnosis"

order by "Expired\_Patient" desc

Table

Description automatically generated

**-- 5. Use a function to calculate the percentage of patients transferred to other hospital per month.**

create or replace function count\_transfer()

returns int

Language PLPGSQL

as $$

declare count\_transfer int;

begin

select count("Patient\_ID") into count\_transfer

from "Discharges" as d

join "DischargeDisposition" as dd on d."Discharge\_ID" = dd."Discharge\_ID"

where dd."DischargeDisposition" = 'Transfer';

return count\_transfer;

end $$;

select date\_part('month',d."DischargeDate") as month,

round(count(d."Patient\_ID")\*100.0/ (select count\_transfer()),2) as percent\_transfer

from "Discharges" as d

join "DischargeDisposition" as dd on d."Discharge\_ID" = dd."Discharge\_ID"

where dd."DischargeDisposition" = 'Transfer'

group by month

order by month

Table

Description automatically generated

**- 6. Calculate the percentage of expired patients in each month.**

**--Here, a function is created which returns the total expired persons and that then the function is called in next set of queries to input the total expired patients to get the percentage.**

create or replace function count\_expired()

returns int

Language PLPGSQL

as $$

declare count\_expired int;

begin

select count("Patient\_ID") into count\_expired

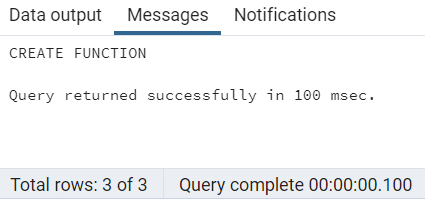
from "Discharges" as d

join "DischargeDisposition" as dd on d."Discharge\_ID" = dd."Discharge\_ID"

where dd."DischargeDisposition" = 'Expired';

return count\_expired;

end $$;



select date\_part('month',d."DischargeDate") as month,

round(count(d."Patient\_ID")\*100.0/ (select count\_expired()),2) as percent\_expired

from "Discharges" as d

join "DischargeDisposition" as dd on d."Discharge\_ID" = dd."Discharge\_ID"

where dd."DischargeDisposition" = 'Expired'

group by month

order by month

Table

Description automatically generated

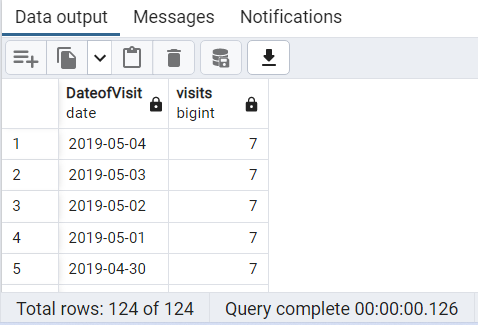
**--7. Get the daily number of patients.**

select "DateofVisit" ,count("Patient\_ID") as visits

from "AmbulatoryVisits"

group by "DateofVisit"

order by "DateofVisit" desc



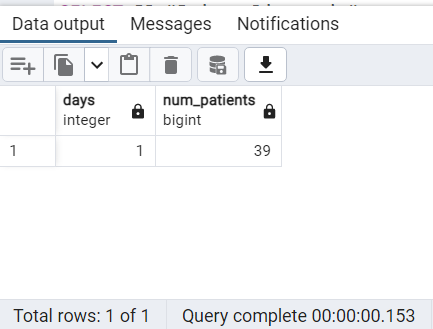
**--8. Number of patients visited the hospital on the next day of scheduled visit.**

select "DateofVisit"-date("DateScheduled") as days, count("Patient\_ID")as num\_patients

from "AmbulatoryVisits"

where "DateofVisit"-date("DateScheduled")=1

group by "DateofVisit"-date("DateScheduled");



**--9. Number of patients admitted in every service.**

SELECT S."Service", COUNT(D."Patient\_ID") as count\_patients

FROM "Discharges" AS D

JOIN "Service" AS S

ON D."Service\_ID" = S."Service\_ID"

GROUP BY S."Service"

order by count\_patients desc

Table

Description automatically generated

**--10. Get a list of top 3 primary diagnosis based on number of patients got readmitted.**

SELECT PD."PrimaryDiagnosis", count(R."Patient\_ID") AS "No\_of\_patients"

FROM "ReAdmissionRegistry" AS R JOIN "PrimaryDiagnosis" AS PD

ON R."Diagnosis\_ID"= PD."Diagnosis\_ID"

GROUP BY PD."PrimaryDiagnosis"

Order by "No\_of\_patients" desc limit 3

Graphical user interface, table

Description automatically generated with medium confidence

**--11. Get the number of patients who have at least two visits to the hospital.**

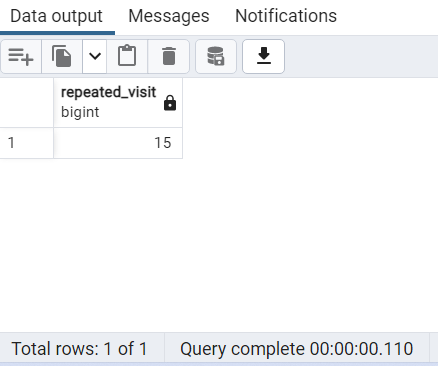
select count(\*) as repeated\_visit from

(select count("Patient\_ID")as count

from "ReAdmissionRegistry"

group by "Patient\_ID" ) x

where x.count>=2



**-- 12. Classify the patients by the birthyear who got admitted to hospital due to heart failure.**

create view heart\_patients as

select p."Patient\_ID","DateOfBirth" from "Patients" as p

join "Discharges" as d on

p."Patient\_ID"=d."Patient\_ID"

join "PrimaryDiagnosis" as pd on

pd."Diagnosis\_ID"=d."Diagnosis\_ID"

where "PrimaryDiagnosis"='Heart Failure';

with ranges as (

SELECT 'Born before 1960' r UNION ALL

SELECT 'Born between 1961 and 1970' UNION ALL

SELECT 'Born between 1971 and 1980' UNION ALL

SELECT 'Born between 1981 and 1990' UNION ALL

SELECT 'Born after 1990'

), setup as (

SELECT CASE

WHEN DATE\_PART('year', "DateOfBirth") < 1961 THEN 'Born before 1960'

WHEN DATE\_PART('year', "DateOfBirth") BETWEEN 1961 AND 1970 THEN 'Born between 1961 and 1970'

WHEN DATE\_PART('year', "DateOfBirth") BETWEEN 1971 AND 1980 THEN 'Born between 1971 and 1980'

WHEN DATE\_PART('year', "DateOfBirth") BETWEEN 1981 AND 1990 THEN 'Born between 1981 and 1990'

WHEN DATE\_PART('year', "DateOfBirth")>1990 THEN 'Born after 1990'

END as yearRange,

"Patient\_ID"

FROM heart\_patients

)

SELECT t.r as Birth\_Range, COUNT(s."Patient\_ID") as Frequency

FROM ranges t

LEFT JOIN setup s

ON t.r = s.yearRange

GROUP BY t.r

order by t.r

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

**--13. Top 5 providers based on the ambulatory visit patients.**

select "ProviderName", count(av."Patient\_ID") as "No\_of\_patients"

from "AmbulatoryVisits" av

join "Providers" p on

av."Provider\_ID" = p."Provider\_ID"

group by "ProviderName"

Order by "No\_of\_patients" desc limit 5

Table

Description automatically generated

**14. List the patients whose pulse rate or BP is either above or below normal.**

select "Patient\_ID","DateofVisit" from "AmbulatoryVisits"

where "BloodPressureSystolic"<90

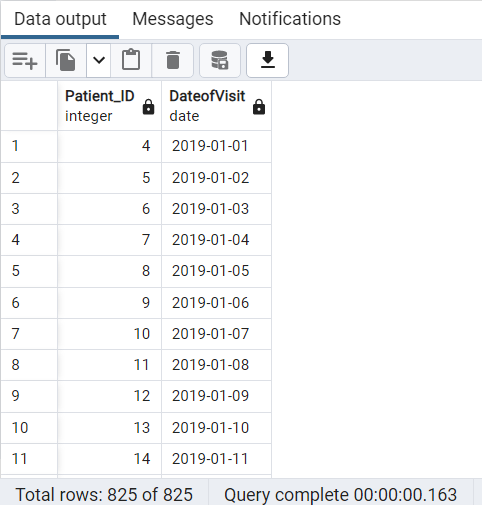
or "BloodPressureSystolic">120

or "BloodPressureDiastolic"<60

or "BloodPressureDiastolic">80

or "Pulse"<60

or "Pulse">100

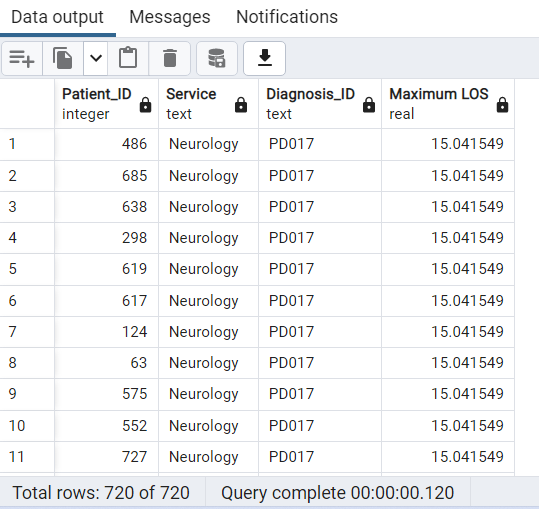


**--15. list of Patient\_ID along with maximum Length of Stay partition by Diagnosis.**

SELECT "Patient\_ID","Service","Diagnosis\_ID",

max("ExpectedLOS") over (Partition by "Diagnosis\_ID" ORDER BY "Diagnosis\_ID" DESC) a

"MaximumLOS" FROM "Discharges" as D join "Service" as S on S."Service\_ID"=D."Service\_ID";



**--16. Number of expired patients in ICU who are below 40 years of age.**

select s."Service", count(d."Patient\_ID") as "No\_of\_Expired patients"

from "Discharges" AS d

JOIN "Service" AS s ON

s."Service\_ID" = d."Service\_ID"

JOIN "DischargeDisposition" AS dd ON

d."Discharge\_ID" = dd."Discharge\_ID"

join "Patients" P on P."Patient\_ID" = d."Patient\_ID"

WHERE dd."DischargeDisposition"='Expired' and DATE\_PART('year',d."AdmissionDate")-DATE\_PART ('year', P."DateOfBirth") <'40' and s."Service" ='ICU'

Group by s."Service"

Graphical user interface, application

Description automatically generated

**--17. Get list of expired patients ID who got admitted due to car accident.**

select d."Patient\_ID", concat("FirstName", ' ',"LastName") as "Fullname", "ReasonForVisit"

from "EDVisits" ed join "ReasonForVisit" rv on

ed."Rsv\_ID"= rv."Rsv\_ID"

join "Discharges" AS d on

d."Patient\_ID"= ed."Patient\_ID"

JOIN "DischargeDisposition" AS dd ON

d."Discharge\_ID" = dd."Discharge\_ID"

join "Patients" p on

P."Patient\_ID"= d."Patient\_ID"

where dd."DischargeDisposition"='Expired' and "ReasonForVisit"= 'Car Accident'

Graphical user interface, table

Description automatically generated with medium confidence

**--18. Number of expired patients based on service.**

select s."Service", count(d."Patient\_ID") as "No\_of\_Expired patients" from "Discharges" AS d

JOIN "Service" AS s ON

s."Service\_ID" = d."Service\_ID"

JOIN "DischargeDisposition" AS dd ON

d."Discharge\_ID" = dd."Discharge\_ID"

WHERE dd."DischargeDisposition"='Expired'

GROUP BY s."Service"

Table

Description automatically generated

**--19. Rank the patients based on Service and Expected Mortality.**

select "Patient\_ID", "ExpectedMortality" ,

"Service", rank() over(partition by ra."Service\_ID" order by "ExpectedMortality" desc ) as "Rank"

from "ReAdmissionRegistry" ra

join "Service" s on ra."Service\_ID" = s."Service\_ID"

Table

Description automatically generated

**--20.Rank the service by using window function based on admissions.**

create or replace view patient\_service as

SELECT count("Patient\_ID") as patients,rr."Service\_ID"

FROM "ReAdmissionRegistry" as rr

join "Service" as S on S."Service\_ID"=rr."Service\_ID"

group by rr."Service\_ID";

select "Service",patients ,

rank() over( order by patients desc ) as "Rank"

from patient\_service as ps

join "Service" as s on

s."Service\_ID"=ps."Service\_ID"

Table

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application, Word

Description automatically generated

**21. Find which Visit type of patients are maximum in cancelling their appointment.**

**-- A subquery is created to get the count of patients for each "AMV\_ID". And then main query is**

**--is written to get the visit type where there are max patients. (Run both the queries together)**

create or replace view visit\_id\_cancelled as

select AV."AMVT\_ID", count (AV."AMVT\_ID") as "Cancelled" from "AmbulatoryVisits" as AV

join "VisitStatus" as VS on AV."VisitStatus\_ID" = VS."VisitStatus\_ID" where VS."VisitStatus" = 'Canceled'

group by AV."AMVT\_ID";

select "VisitType" from "VisitTypes" as vt join visit\_id\_cancelled as vic on vt."AMVT\_ID"=vic."AMVT\_ID"

where vic."Cancelled"=(select max ("Cancelled") from visit\_id\_cancelled );

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated